Mmg3d : User Guide

Version 1.0

November 30, 2011

Contents

1 Mesh generation by local modifications **2**
- 1.1 Context . . . 2
- 1.2 Overview . . . 2

2 Installation **3**
- 2.1 Distribution and authors . . . 3
- 2.2 Langages and platforms . . . 3
- 2.3 Compilation and installation . . . 3
  - 2.3.1 Compilation . . . 3
  - 2.3.2 Installation . . . 3

3 Mesh generation using binary **3**
- 3.1 Input . . . 3
- 3.2 Options . . . 5
- 3.3 Running . . . 6

4 Mesh generation with library using **6**
- 4.1 File arguments . . . 6
- 4.2 Options . . . 6
- 4.3 Mesh format . . . 7
- 4.4 Metric definition/initialization . . . 9
- 4.5 Out . . . 10

5 Bodies movement **10**
- 5.1 Data input . . . 10
- 5.2 Options . . . 10
- 5.3 Running . . . 10
- 5.4 Data output . . . 10

6 Hexahedra/Prisms cutting **11**
- 6.1 Data input/output . . . 11
- 6.2 Hexahedra cutting . . . 11
- 6.3 Prims cutting . . . 11

7 Results evaluation **11**
- 7.1 Elements quality . . . 11
- 7.2 Edges lengths and efficiency index . . . 12

8 Mesh visualization **12**

9 Error and warning **12**

10 Examples **13**
- 10.1 Mesh adaptation . . . 13
  - 10.1.1 Uniform metric aligned with axes . . . 13
  - 10.1.2 Anisotropic metric . . . 13
- 10.2 Rigid bodies moving . . . 15

11 FAQ **17**

A Appendix 1 : Files format (MESH and SOL) **17**

1 Mesh generation by local modifications

1.1 Context

The aim of the local mesh methods is to modify iteratively a mesh with local operators: points insertion, edges supression, edge swaps... Those methods can be used to refine/derefine a mesh, improve the elements form (quality) or control the edges length.

The local aspect allows to extend them easilly to rigid bodies problems and moving meshes.

1.2 Overview

Generalities: `Mmg3d` is an anisotropic fully tetrahedral automatic remesher. Beginning with a tetrahedral mesh, it modifies it iteratively to be in agreement with prescribed sizes (and directions) and/or to move all or a part of the domain boundary (rigid bodies displacement).

The software reads a mesh and it is possible to prescribe a map for edges sizes/directions and/or a displacement vector for the mesh nodes.

The mesh is then modifying while using local mesh modification of which an insertion procedure based on anisotropic Delaunay kernel.

NB : For the moment, only volumic mesh (i.e. the tetrahedra) is treated. No modification of the surface triangulation made (except for the rigid bodies movements and global splitting).

Mesh adaptation/optimization: Starting from a mesh and a size/direction map, the mesh is modified to fit with the prescription. The algorithm is decomposed in two parts. First, the edges lengths are analysed: the long edges are cutted and the short are deleted. Then, when the point density are in agreement with the prescribed size, there is a phase of quality optimization (edges swaps and point relocation).

NB : In the case where there is no prescribed size, we create one with the initial mesh (in analysing the edges lenghts).

Bodies movement: When a displacement is prescribed on the initial mesh, the final mesh will be such as the surface vertex will be at a new position corresponding with the prescribed displacement. This movement can be made with keeping constant the number of nodes (but it is not an obligation). The final mesh will always be valid (in conform meaning).

NB : The surface vertex are moved without geometric checks. All quality criterion are based on volumic mesh.

2 Installation

2.1 Distribution and authors

Mmg3d is written by two authors:

Cécile DOBRZYNSKI
IMB, Institut Polytechnique de Bordeaux
INRIA Bordeaux - Sud-Ouest, EPI Bacchus
351, cours de la Libération
33405 Talence cedex
cecile.dobrzynski@math.u-bordeaux1.fr

Pascal FREY
Laboratoire Jacques Louis Lions
Université Pierre et Marie Curie
4, place Jussieu, tour 15-25
75005 Paris
frey@ann.jussieu.fr

Version 4.0 of Mmg3d is distributed as free/libre software, under the terms of the GNU GENERAL PUBLIC LICENSE. The term of this license is detailled here:
http://www.math.u-bordeaux1.fr/~dobj/logiciels/download/LICENCE.txt

You can download the source code of Mmg3d as well as precompiled version on the following website:
http://www.math.u-bordeaux1.fr/~dobj/logiciels/download/

2.2 Langages and platforms

Mmg3d is entirely written in C (C89 ANSI norm). The current version counts about 55000 lines of source code. The code is optionnaly coupled with SCOTCH library [] and has been tested on all major computer architectures and operating systems.

2.3 Compilation and installation

2.3.1 Compilation

To easily compile Mmg3d , please use cmake tools [] and refer to the file INSTALL.txt.

Remark: By default, Mmg3d use the library SCOTCH to renumbering the node and the element of the initial mesh. You can download this library on the following website: http://www.labri.fr/perso/pelegrin/scotch/

If you don't want use SCOTCH, you can specify on cmake that you would compile without this library. In this case, Mmg3d could run more slowly.

2.3.2 Installation

If you download precompiled version of Mmg3d , you just have to unzip it: gzip -d mmg3d4.0-linux.gz

3 Mesh generation using binary

3.1 Input

- The input mesh will be to mesh format. This format is just composed by one file (binary or ASCII), for example toto.mesh ou toto.meshb.

  This file is organised in series of fields identified by key words. Besides vertex coordinates and tetrahedra list, you can specify other information like faces list, corners, edges, etc.

Example 1:

```
#Example MESH file with triangles and tetrahedra
MeshVersionFormatted 2

Dimension 3

Vertices
8
0 0 0 1
0 1 0 1
1 1 0 1
1 0 0 1
0 0 1 1
0 1 1 1
1 1 1 1
1 0 1 1

Triangles
12
1 4 8 15
1 2 4 10
3 2 7 14
8 3 7 12
5 8 6 11
2 5 6 13
5 2 1 13
8 5 1 15
7 6 8 11
2 6 7 14
4 3 8 12
2 3 4 10

Tetrahedra
6
1 2 8 4 2
8 3 2 7 2
2 5 8 6 2
8 5 2 1 2
2 7 6 8 2
2 4 3 8 2

End
```

Example 2:

```
#Example MESH file with quadrilaterals and hexahedra

MeshVersionFormatted 2

Dimension 3

Vertices
8
0 0 0 1
0 1 0 1
1 1 0 1
1 0 0 1
0 0 1 1
0 1 1 1
1 1 1 1
1 0 1 1

Quadrilaterals
6
1 2 3 4 10
5 6 7 8 11
4 3 7 8 12
1 2 6 5 13
3 2 6 7 14
4 1 5 8 15

Hexahedra
1
1 2 3 4 5 6 7 8 2

End
```

The detailled description of this format is made in appendix A.

- An isotropic/anisotropic metric can be specify on each input mesh vertices. This metric is given in an other file with SOL format (binary or ASCII). This file must have the same name as the MESH file with the extension `.sol` (for example `toto.sol` or `toto.solb`). This file is organised in series of fields identified by key words.

Example 1:

```
#Example SOL file for isotropic metric
MeshVersionFormatted 2

Dimension 3

SolAtVertices
8
1 1
0.1
0.1
0.1
0.1
0.1
0.1
0.1

End
```

Example 2:

```
#Example SOL file for anisotropic metric
MeshVersionFormatted 2

Dimension 3

SolAtVertices
8
1 3
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100
100 0 100 0 0 100

End
```

The detailled description of this format is made in appendix A.

3.2 Options

All options are given in command line:

- `-O` n with n an integer:

`-O 0`	:	mesh optimization.
`-O 1`	:	mesh generation in agreement with a prescribed metric, including points insertion/suppression and mesh optimization.
`-O 4`	:	option 1 + global splitting, warning global splitting involves splitting of boundaries (triangles)
`-O 9`	:	rigid bodies displacement and optimization (see 5).
`-O 10`	:	transform an hexahedral/prism mesh in tetrahedral one (see 6).
`-O -1`	:	same as described previously without the optimization phase.

- `-d` : debug mode.
- `-h` : command summary, the following message will be written in your xterm:

```
   -- MMG3d, Release 4.0 c (July 20, 2010)
      Copyright (c) LJLL/IMB, 2010
      Lun 4 avr 2011 11:23:20 CEST
usage: mmg3d4.0 [-v[n]] [-h] [-m n] [opts..] filein[.mesh] [-out fileout]

** Generic options :
-d      Turn on debug mode
-h      Print this message
-v [n]  Tune level of verbosity
-m [n]  Set memory size to n Mbytes

-O [n]  Optimization level
  1      adaptation
  4      use global splitting (warning modify boundary mesh)
  9      moving mesh
  10     transform an hexahedral/prism mesh in a tetrahedral mesh
 -n      turn off optimisation

** Misc. options
-bucket [n]    Specify the size of bucket per dimension
-noswap        no edge or face flipping
-nomove        no point relocation
-noinsert      no new point
-out fileout   Specify output file name
-rn n num         Specify the number of vertices by box to renumber nodes and the renumberings
-dt dt         to compute the node speed
```

- `-v` n
  n defined the verbosity level (defaulft 3).

- `-m` n
  the integer n defined the allocated memory in Megabytes.
  By default, the software allows to generate a mesh with maximum 500 000 points or 3 000 000 tetrahedra.

- `-mov name` : gives the displacement filename (option 9, see 5).

- `-dt t` : to compute the node speed (option 9, see 5).

- `-in name` : gives the input filename.

- `-out nom` : gives the output filename.

- `-bucket` : specifies the bucket size (default 64).

- `-noswap` : generate the mesh without applied edge swaps.

- `-noinsert` : generation the mesh in keeping constant the number of nodes (no points insertion or edge deletion).

- `-nomove` : generate the mesh without point relocation in optization stage.

- `-rn` *n num* : this option concerns renumbering stage. The integer *n* is the number of vertices by box (by default 500) and *num* allows to tell when the renumbering stage appears: 0: no renumbering stage ; 1: renumbering at beginning ; 2: renumbering after insertion stage ; 3 (default): renumbering at the beginning and after insertion stage.

3.3 Running

In this paragraph, we suppose that you have a MESH file named `c1`.mesh (http://www.math.u-bordeaux1.fr/~dobj/logiciels/data/c1.mesh) that you would remesh and optionnaly an associated metric named `c1`.sol (http://www.math.u-bordeaux1.fr/~dobj/logiciels/data/c1.sol). To launch the software, it is enough to type one of the following commands in terminal window:
`mmg3d4.0 -O 1 c1` or `mmg3d -O 1 -in c1`

You will obtain a new mesh named `c1.o.meshb` and the associated metric named `c1.o.solb`.
If you would modify the output name: `mmg3d4.0 -O 1 c1 myoutname.mesh` or `mmg3d -O 1 -in cube -out myoutname.mesh`
In this case, you will obtain the two following files : `myoutname.mesh` and `myoutname.sol`

To visualize this mesh, you could use the software `Medit` downloadable at the following url:
http://www.ann.jussieu.fr/~frey/logiciels/

Remark : in the case where there is no metric file, the software build one starting from the initial mesh.

NB : if you do not give the file extension (.mesh(b)), `Mmg3d` first find the binary file then if there is no meshb file, it tries to open ASCII file. In other words, if you have in the same directory the files `toto.mesh` and `toto.meshb` and that you type `mmg3d -O 1 toto`, the infile will be `toto.meshb`.

4 Mesh generation with library using

4.1 File arguments

To use `Mmg3d` as a library, you must include the file `libmmg3d.h` and call the following `C`-function:

```
int MMG_mmg3dlib(int opt[7],MMG_pMesh mesh,MMG_pSol sol)
```

This function takes three arguments:

1. a tabular containing nine integers. This tabular allows to specify code option, see 4.2,
2. a pointer on a `MMG_Mesh` object which contains initial mesh informations, see 4.3,
3. a pointer on a `MMG_sol` object which contains metric specification , see 4.4.

This function returns an integer which is strictly positive if the mesh procedure anormally ended. At the end, the pointers of `MMG_Mesh` and `MMG_Sol` types contain the new mesh and the associated metric.

4.2 Options

As on the binary using, the `Mmg3d` call is done with options. This one must be specified on the tabular `int opt[9]`. This tabular contains:

0. an integer defining the remeshing mode:

`0`	:	mesh optimization.
`1`	:	mesh generation in agreement with a prescribed metric, including points insertion/suppression and mesh optimization.
`4`	:	option 1 + global splitting, warning global splitting involves splitting of boundaries (triangles)
`9`	:	rigid bodies displacement and optimization (see 5).
`10`	:	transform an hexahedral/prism mesh in tetrahedral one (see 6).
`-1 or -4 or -9`	:	same as described previously without the optimization phase.

1. debugging mode: 1 for debugging node, 0 if not,

2. bucket size: default 64,

3. the possibility to generate the mesh with applied or not edge swaps: 1 not authorizing edge swaps.

4. the possibilty to generate the mesh in keeping constant the number of nodes (no points insertion or edges supression): 1 to keep constant the number of nodes, 0 if not,

5. the possibilty to generate the mesh with or without point relocation in optimization stage: 1 to avoid point relocation, 0 if not,

6. verbosity level: default 3,

7. this integer concerns renumbering stage, it allow to tell when renumbering stage appears:
   0: no renumbering stage ; 1: renumbering at beginning ; 2: renumbering after insertion stage ; 3 (default): renumbering at the beginning and after insertion stage.

8. this integer concerns renumbering stage, it is the number of vertices by box (default 500),

4.3 Mesh format

To use `Mmg3d` as a library, you must define the informations describing the mesh. This informations are specified in a particular structure named `Mesh` (the prototype is in `libmmg3d.h`).

Details for the structure `MMG_Mesh` : This structure can be decomposed on five parts :

1. the general mesh informations:
   number of nodes (`np`) and maximal number of nodes which will be generated (`npmax`)
   number of surface triangles (`nt`) and maximal number of triangles which could be generated (`npmax`)
   number of tetrahedra (`ne`) and maximal number of tetrahedra which could be generated (`nemax`)

2. the vertex (`MMG_pPoint point`):
   This part contains for all the vertex, their coordonnates and their logic (reference)

3. the surface triangles (`MMG_pTria tria`):
   This part contains for all the surface triangles, the points number of their three vertex and their logic (reference),

4. the tetrahedra (`MMG_pTetra tetra`):
   This part contains for all the tetrahedra, the points number of their four vertex and their logic (reference, number of subdomain),

5. the displacement prescribed on each vertex (in rigid movement case, see 5)(`MMG_pDispl disp`):
   This part contains a displacement vector for all vertex mesh.

Definition/initialisation of MMG_Mesh structure: to define the MMG_Mesh structure, you need to allocate and initialize all the elements described on the previous paragraph. The next steps explains how to do that and dive the C-code corresponding:

1. Define a pointer on a MMG_Mesh structure:
   MMG_pMesh mymmgmesh

2. Allocate this pointer:
   mymmgmesh = (MMG_pMesh)calloc(1,sizeof(MMG_Mesh));

3. Give the nodes, triangles and tetrahedra numbers of your mesh:
   mymmgmesh->np = nbpoint;
   mymmgmesh->nt = nbtri;
   mymmgmesh->ne = nbtetra;

4. Give the nodes, triangles, tetrahedra maximum numbers:
   mymmgmesh->npmax = nbpointmax;
   mymmgmesh->ntmax = nbtrimax;
   mymmgmesh->nemax = nbtetramax;

5. Allocate the structure point, tetra, tria, disp et adja

```
mymmgmesh->point = (MMG_pPoint)calloc(mymmgmesh->npmax+1,sizeof(MMG_Point));
mymmgmesh->tetra = (MMG_pTetra)calloc(mymmgmesh->nemax+1,sizeof(MMG_Tetra));
mymmgmesh->tria  = (MMG_pTria) calloc(mymmgmesh->ntmax+1,sizeof(MMG_Tria));
mymmgmesh->adja  = (int*)calloc(4*mymmgmesh->nemax+5,sizeof(int));
mymmgmesh->disp  = (MMG_pDispl)calloc(mymmgmesh->npmax+1,sizeof(MMG_Displ));

mymmgmesh->disp->mv  = (double*)calloc(3*(mymmgmesh->npmax+1),sizeof(double));
mymmgmesh->disp->alpha  = (short*)calloc(mymmgmesh->npmax+1,sizeof(short));
```

6. Initialize the point structure, that means for all the nodes, give the coordinates and the reference:

```
MMG_pMesh mymmgmesh;
MMG_pPoint  ppt;

 for (k=1; k<=mymmgmesh->np; k++) {
   ppt = &mymmgmesh->point[k];
   ppt->c[0] = coorx;
   ppt->c[1] = coory;
   ppt->c[2] = coorz;
   ppt->ref  = logic;
 }
```

   NB : The node are numbered from 1 to mymmgmesh->np in a contiguous way.

7. Initialize the tetra structure, that means for all tetrahedra, give the number nodes of the four vertex and the reference:

```
MMG_pMesh mymmgmesh;
MMG_pTetra  ptetra;

 for (k=1; k<=mymmgmesh->ne; k++) {
   ptetra = &mymmgmesh->tetra[k];
```

```
  ptetra->v[0] = num1;
  ptetra->v[1] = num2;
  ptetra->v[2] = num3;
  ptetra->v[3] = num4;
  ptetra->ref  = logic;
}
```

NB : The tetrahedra are numbered from 1 to `mymmgmesh->ne` in a contiguous way.

8. (optional) Initialize the `tria` structure, that means for all surface triangles, dive the node number of the three vertex and the reference.

```
MMG_pMesh mymmgmesh;
MMG_pTria ptriangle;

 for (k=1; k<=mymmgmesh->nt; k++) {
   ptriangle = &mymmgmesh->tria[k];
   ptriangle->v[0] = num1;
   ptriangle->v[1] = num2;
   ptriangle->v[2] = num3;
   ptriangle->ref  = logic;
 }
```

NB : The node are numbered from 1 to `mymmgmesh->nt` in a contiguous way

9. (optional) If you treat, bodies movement, initialize the `disp` structure:

```
MMG_pMesh mymmgmesh;
MMG_pDispl  pd;

  for (k=1; k<mymmgmesh->np; k++) {
    pd  = &mesh->mymmgdisp[k];
    pd->mv[0] = depx;
    pd->mv[1] = depy;
    pd->mv[2] = depz;
 }
```

4.4 Metric definition/initialization

To adapt the mesh according to a metric, you must define it in a `MMG_Sol` structure, like that:

1. Define a pointer on a `Sol` structure:
   `MMG_pSol sol;`

2. Allocate this pointer:
   `sol = (MMG_pSol)calloc(1,sizeof(MMG_Sol));`

3. Give the points number:
   `sol->np = nbpoint;` (this number is equal to `mesh->np`)
   `sol->npmax = nbpointmax;` (this number is equal to `mesh->npmax`)

4. Give the metric type, *i.e.* isotropic or anisotropic:
   `sol->offset = 6` for an anisotropic metric
   `sol->offset = 1` for an isotropic metric

5. Allocate the tabular containing the metric field:
```
sol->met = (double*)calloc(sol->npmax+1,sol->offset*sizeof(double));
sol->metold = (double*)calloc(sol->npmax+1,sol->offset*sizeof(double));
```

6. Initialize the `met` structure that means give on each point the metric values:

```
for (k=1; k<=mesh->np; k++) {
  isol = (k-1) * sol->offset + 1;
  for (i=0; i<sol->offset; i++)
    sol->met[isol + i] = metrique[ pointindice_k + i ];
}
```

4.5 Out

The `MMG_pMesh` and the `MMG_pSol` returned by the function are respectively the final mesh and the final metric.

5 Bodies movement

5.1 Data input

Same as the standard case 3.1, the software reads a mesh format file containing the initial mesh and possibly a sol file containing the metric field.

To do the mesh movement, the software reads a second sol file containing a displacement vector defining on each mesh vertex. However, only the displacements on boundary vertex are taken into account in `Mmg3d` .

5.2 Options

All options are specified on command line. See 3.2.

5.3 Running

In this paragraph, we suppose that you have a mesh file named `c1.mesh` that you would remesh (http://www.math.u-bordeaux1.fr/~dobj/logiciels/data/c1.mesh) , a sol file containing the displacement on each mesh vertex `dep.sol` and optionnaly an associated metric named `c1.sol` (http://www.math.u-bordeaux1.fr/~dobj/logiciels/c1.sol). To launch the software, it is enough to type the following command in terminal window: `mmg3d -O 9 cube -mov dep`

You will obtain a new mesh which the name will be `cube.o.meshb`.
Remark : to obtain an ASCII file, you have to specify the output name: `mmg3d -O 9 cube cube.o.mesh -mov dep`

To visualize this mesh, you could use the software `Medit` downloadable at the following url:
http://www.ann.jussieu.fr/~frey/logiciels/

Remark : in the case where there is no metric file, the software build one starting from the initial mesh.

NB : if you do not give the file extension (.mesh(b)), `Mmg3d` first find the binary file then if there is no mesh(b) file, it tries to open ASCII file. In other words, if you have in the same directory the files `toto.mesh` and `toto.meshb` and that you type `mmg3d -O [n] toto`, the infile will be `toto.meshb`. To use the file `toto.mesh`, you have to type `Mmg3d -O [n] toto.mesh`

5.4 Data output

After running, you will obtain two files : a mesh file containing the new mesh and a sol file.

6 Hexahedra/Prisms cutting

6.1 Data input/output

The software reads a mesh format file (see A) containing the initial mesh. This mesh can be composed by hexahedra, prims and tetrahedra and boundary elements can be composed by triangles and/or quadrilaterals. After running, you will obtain a mesh file containing tetrahedra and triangles.

6.2 Hexahedra cutting

A majority of hexadra are cutted into six tetrahedra when it is possible. The others are subdivised into twelve hedrahedra with adding a vertex at the barycenter of the hexadron.

6.3 Prims cutting

A majority of hexadra are cutted into three tetrahedra when it is possible. The others are subdivised into eight tetrahedra with adding a vertex at the barycenter of the prims.

7 Results evaluation

7.1 Elements quality

Definition: The quality measurement for a tetrahedron K on isotropic case is the following:

$$Q_K = \frac{h_s^3}{V_K},$$

where $h_s = \sqrt{\sum_{i=1}^{6} h_i^2}$, h_i being the length of the edge i of the tetrahedron K and V_K its volume.

When an anisotropic metric is defined on each mesh point, the quality is more difficult to define. In `Mmg3d` , we choose to calculate the tetrahedron quality as its quality measured on the euclidian space relative to an average metric defined on the tetrahedron. Lets P_1, P_2, P_3, P_4 the vertex of K. The average quality on K is given by the following formula:

$$\mathcal{M}_{moy} = \frac{1}{4}\left(\sum_{i=1}^{4} \mathcal{M}_i^{-1}\right)^{-1},$$

where $\mathcal{M}_i$ is the metric at P_i.

The following expression give the quality of K on the metric $\mathcal{M}_{moy}$:

$$Q_K = \frac{\left(\sum_{1\leq i<j\leq 6} {}^t\overrightarrow{P_iP_j}\,\mathcal{M}_{moy}\,\overrightarrow{P_iP_j}\right)^{\frac{3}{2}}}{\sqrt{Det(\mathcal{M}_{moy})}\,V_K}$$

NB : A such criterion allows to decide if a simplex is a good element. You can notice that the quality Q_K varies in $[1\,,+\infty[$ and that more the volume of the tetrahedra is small and more Q_K will be high.

Histogram: At the end of the execution, `Mmg3d` prints quality histogram. For instance:

```
-- MESH QUALITY   1136
   AVERAGE QUALITY              1.4413
   BEST  ELEMENT QUALITY        1.0767
   WORST ELEMENT QUALITY        2.2574
   WORST ELEMENT   1183 (1125)   277 165 114 224

   HISTOGRAMM
       1 < Q <     2      1130    99.47 %
       2 < Q <     3         6     0.53 %
```

In the previous histogram:

`1136`	number of tetrahedra
`WORST ELEMENT 1183 (1125)`	1125 is the number of the tetrahedra with the worst quality

7.2 Edges lengths and efficiency index

Definition: We define an index allowing to quickly judge if the mesh is in agreement with the metric field. Lets note l_i the length of the edge e_i in the metric. The *efficiency index* of a mesh is defined as the exponentiel of the mean between the difference with 1 of all the lengths l_i normalized by the number of mesh edges. Lets call τ this index:

$$\tau = \exp\left(\frac{1}{ne}\sum_{i=1}^{ne} e_i\right)$$

with $e_i = l_i - 1$ si $l_i < 1$ and $e_i = \dfrac{1}{l_i} - 1$ if $l_i > 1$ and ne the number of mesh edges.

Histogram: At the end of the execution, the software prints the efficiency index and a length histogram. For example:

```
-- RESULTING EDGE LENGTHS  1115
   AVERAGE LENGTH               0.9428
   SMALLEST EDGE LENGTH         0.5975      172    252
   LARGEST  EDGE LENGTH         1.5942      215    254
   EFFICIENCY INDEX             0.8630
   0.71 < L < 1.41      1088   97.58 %

   HISTOGRAMM
   0.50 < L < 0.71        18    1.61 %
   0.71 < L < 0.90       554   49.69 %
   0.90 < L < 1.11       330   29.60 %
   1.11 < L < 1.41       204   18.30 %
   1.41 < L < 2.00         9    0.81 %
```

In the previous histogram:

`1115`	number of edges
`172 252`	vertex of the smallest edges
`215 254`	vertex of the largest edges

8 Mesh visualization

To visualize this mesh, you could use the software `Medit` downloadable at the following url:
http://www.ann.jussieu.fr/~frey/logiciels/

9 Error and warning

The principal errors are the following:

- `NOT FOUND.` ou `UNABLE TO OPEN`: you have a problem with the input files.
- `INVALID METRIC`: the specified metric on several points is not a metric: the matrix determinant or the eigenvalues are negatives.
- `WARNING: WRONG SOLUTION NUMBER. IGNORED`: the specified metric is ignored, the file not containing the same number of points than the mesh file.

- `WARNING: INCOMPLETE MESH`: the software did not have enough memory to generate the mesh, thus the final mesh is probably not satured in term of number of nodes. However, the final mesh is valid so you can rerun `Mmg3d` beginning with this final mesh by increasing the memory (`-m 500` for instance).

  NB : In this case, the final mesh is correct (in sense of conformity) even if it is not well adapted to the prescribed metric.

10 Examples

10.1 Mesh adaptation

10.1.1 Uniform metric aligned with axes

The metric tensor is diagonal and defined in following way:

$$\mathcal{M}(P) = \begin{pmatrix} \frac{1}{\alpha^2 h^2} & 0 & 0 \\ 0 & \frac{1}{\beta^2 h^2} & 0 \\ 0 & 0 & \frac{1}{\gamma^2 h^2} \end{pmatrix}$$

The edges length in the direction x (resp. y et z) is $\alpha\, h$ (resp. $\beta\, h$ et $\delta\, h$)

The initial mesh can be one of them represented on the Figure 1. The Figure 2 shows some volumic cuts on adapted meshes for several α, β, δ values.

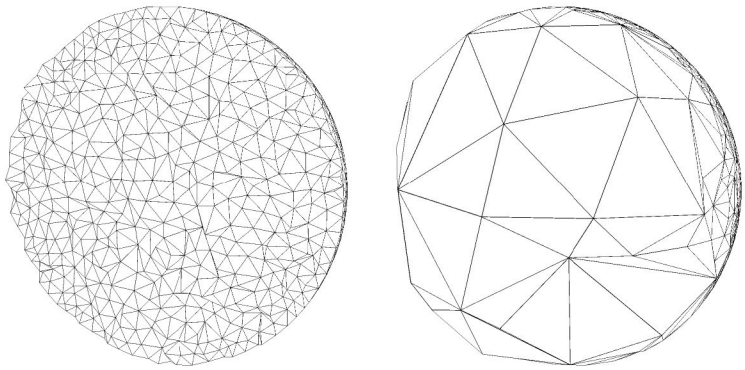

Figure 1: Volumic cuts in initial meshes.

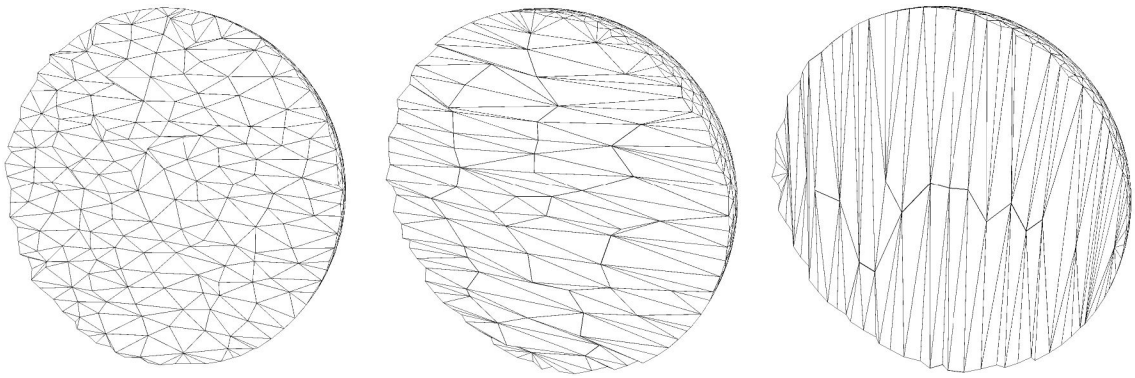

Figure 2: Volumic cut in adapted meshes: in the left top $\alpha = 2$, $\beta = 1$ et $\gamma = 1$; in the right top $\alpha = 1$, $\beta = 5$ et $\gamma = 1$; on the bottom $\alpha = 1$, $\beta = 1$ et $\gamma = 10$.

10.1.2 Anisotropic metric

The metric is defined by a 3×3 symetric positive matrix. The Figure 3 shows an example of mesh adaptation where the metric is analytically defined. An example of anisotropic adaptation on CFD simulation is represented on the Figure 4

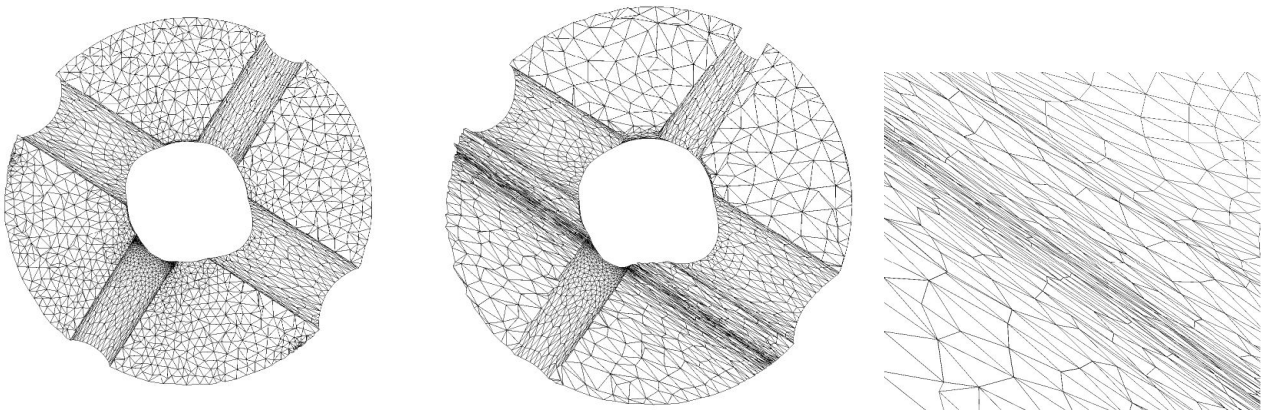

Figure 3: Volumic cuts on initial mesh and adapted mesh. Zoom on anisotropic meshes.

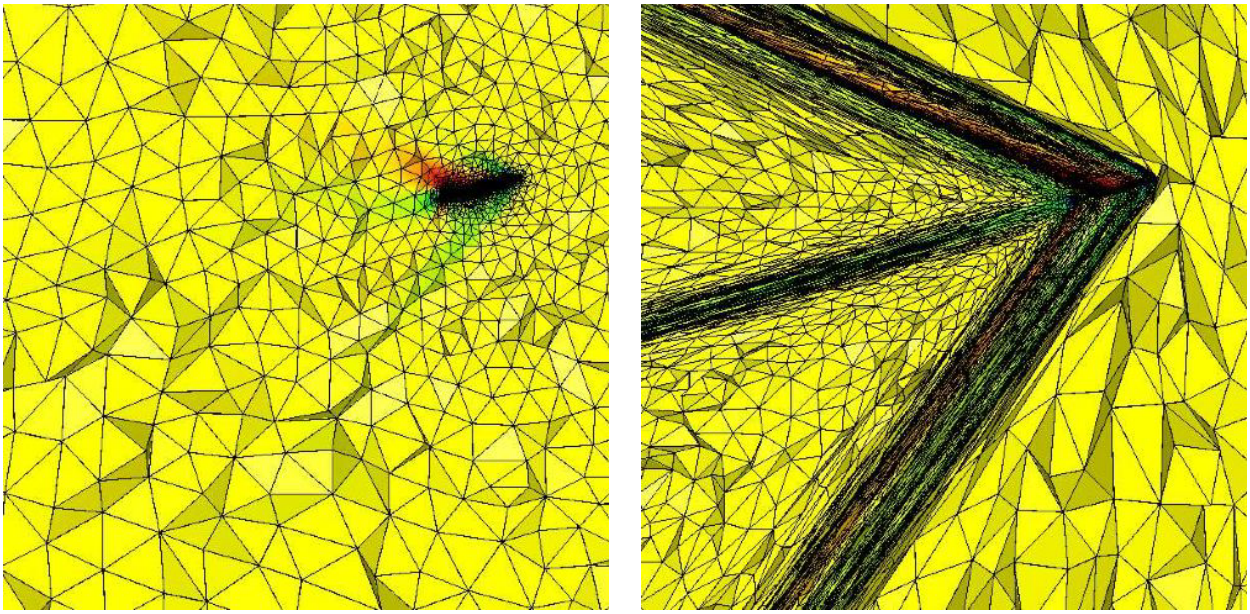

Figure 4: supersonic flow around an airplane (mach 1.6, angle attack 3 degre, diameter of the box 1km): volumic cut on initial and adapted meshes.

10.2 Rigid bodies moving

Moving object on an uniform mesh. The aim of this example is to move horizontaly a car profile on a domain with keeping constant the number of nodes. The Figure 5 shows some volumic cuts on the tetrahedral mesh at three differents steps.

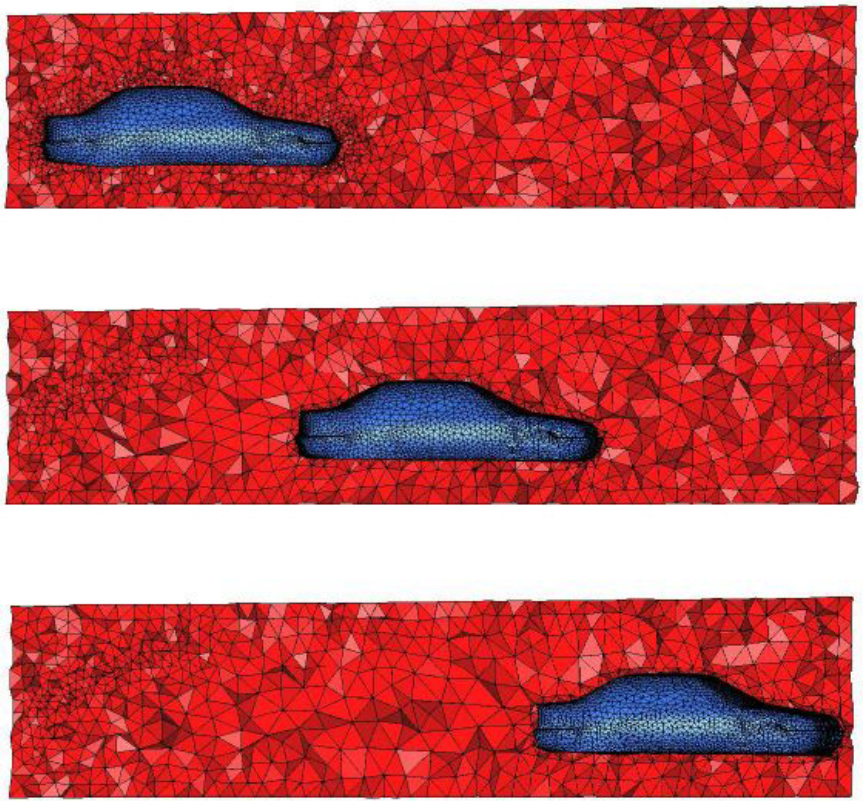

Figure 5: Volumic cuts after one, 75 and 150 boundary moves.

Moving an object on an anisotropic mesh. In this example, the initial mesh is adapted to an anisotropic metric field. The aim is to move two spheres in the domain while keeping the anisotropy. On the Figure 6, some volumic cuts are represented.

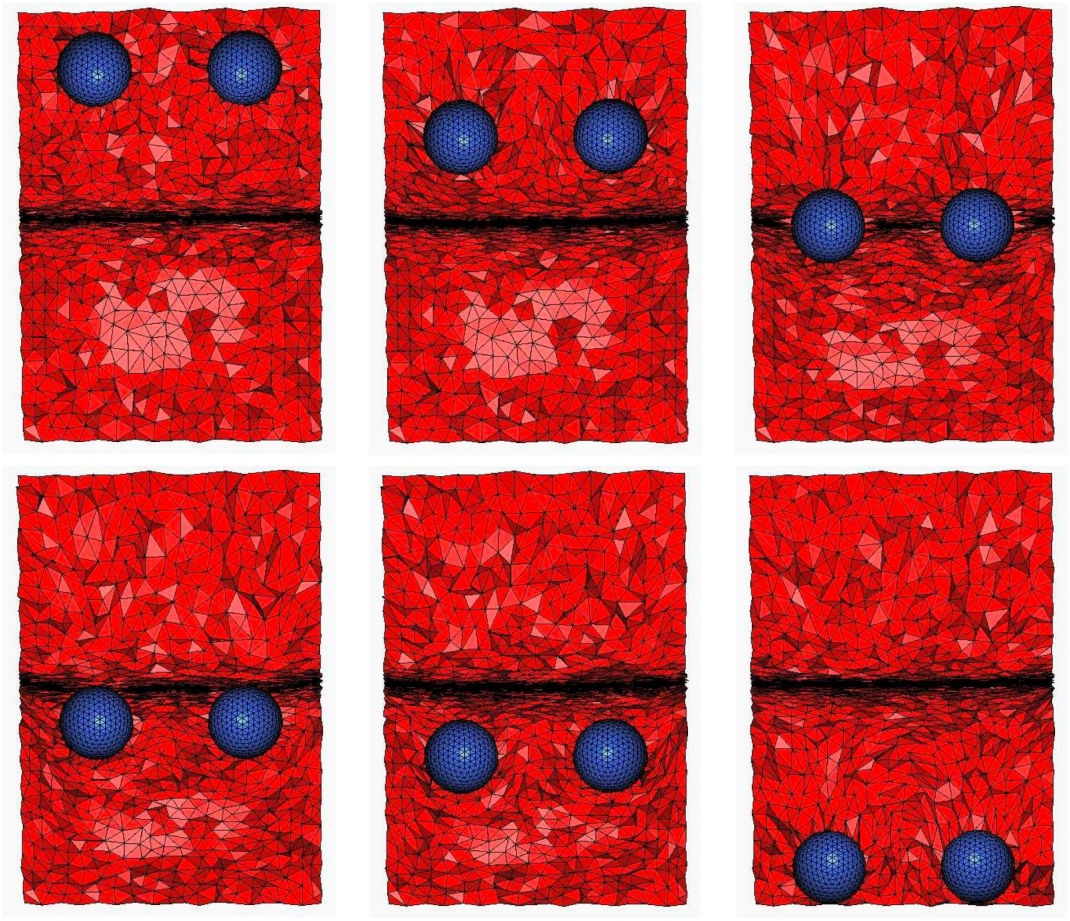

Figure 6: Displacement of 2 particles with respect of an anisotropic metric field (iteration 0, 5, 13, 16, 19).

11 FAQ

- How can I specify Boundary Conditions that remains untouched by `Mmg3d` ?

  The boundary faces are the Triangles. To locate a boundary condition, you can put the same reference on a set of triangles. This reference will be unchanged by `Mmg3d` .

- What must I put like the reference number?

  You can put every number. However the new vertex created by `Mmg3d` will have a reference number equal to zero. If there are several reference numbers for tetrahedra, `Mmg3d` will consider several subdomains.

- In a mesh with only tetrahedra, is it mandatory to specify Edges and Triangles to `Mmg3d` ?

  No, you can give only the tetrahedra.

A Appendix 1 : Files format (MESH and SOL)

`Mmg3d` reads and writes mesh files with `MESH` format and solutions (metric or displacement) with `SOL` format.

Those files formats allows to read/write binary and ASCII files. They are composed by one file (`xx.mesh` or `xx.meshb` or `xx.sol` or `xx.solb`) which contains all needed information for mesh (or solution) description.

On the following paragrah, we will describe first the ASCII format then the binary one.

ASCII file The file is organised with key words. The comments line must beginning with the character #. The "blank", the line return (or `<CR>`) and the "tabs" are considered like separators.

1. The file must beginning by the keyword `MeshVersionFormatted` following with the version number. This number indicates if the file containing single or double precision reals.

   `MeshVersionFormatted` 1 : file writing with simple precision
   `MeshVersionFormatted` 2 : file writing with double precision

2. The next keyword is about the space dimension:

   `Dimension` 3

   NB : the dimension could be equal to 2 but it is not relevant for `Mmg3d` .

3. The file must end by the keyword `End`.

File `MESH` Before the keyword `End`, you have to define all the mesh characteristics: vertex, elements (triangles, quadrilaterals, tetrahedra, hexahedra...), etc. Those entities are defined as follow:

NB : The entities number begins to 1 (and not 0 like in C langage)

`Vertices`
np
$x_1\ y_1\ z_1\ ref_1$
...
$x_{np}\ y_{np}\ z_{np}\ ref_{np}$
with np the number of mesh nodes, $x_i\ y_i\ z_i$ the coordinates of vertex i and ref_i its reference (an integer).
NB : In dimension 2, it is just $x_i\ y_i\ ref_i$

`Triangles`

nt
$v_1^1\ v_1^2\ v_1^3\ ref_1$
...
$v_{nt}^1\ v_{nt}^2\ v_{nt}^3\ ref_{nt}$
with nt the number of mesh triangles, v_i^j the number of vertex j of triangle i and ref_i its reference.

Quadrilaterals
nq
$v_1^1\ v_1^2\ v_1^3\ v_1^4\ ref_1$
...
$v_{nq}^1\ v_{nq}^2\ v_{nq}^3\ v_{nq}^4\ ref_{nq}$
with nq the number of quadrilaterals, v_i^j the number of vertex j of quadrilaterals i and ref_i its reference.

Tetrahedra
$ntet$
$v_1^1\ v_1^2\ v_1^3\ v_1^4\ ref_1$
...
$v_{ntet}^1\ v_{ntet}^2\ v_{ntet}^3\ v_{ntet}^4\ ref_{ntet}$
with $ntet$ the number of tetrahedra, v_i^j the number of vertex j of tetrahedron i and ref_i its reference.

Hexahedra
nh
$v_1^1\ v_1^2\ v_1^3\ v_1^4\ v_1^5\ v_1^6\ v_1^7\ v_1^8\ ref_1$
...
$v_{nh}^1\ v_{nh}^2\ v_{nh}^3\ v_{nh}^4\ v_{nh}^5\ v_{nh}^6\ v_{nh}^7\ v_{nh}^8\ ref_{nh}$
with nh the number of hexahedra, v_i^j the numver of vertex j of hexahedron i and ref_i its reference.

Edges
ne
$e_1^1\ e_1^2\ ref_1$
...
$e_{ne}^1\ e_{ne}^2\ ref_{ne}$
with ne the number of (boundary) edges, $e_i^1 e_i^2$ the numbers of the two vertex which compose the edge i and ref_i its reference.

SOL file Like MESH format, SOL format is composed by keywords describing if the solution is defined on the vertices, triangles, tetrahedra, etc..

NB : On Mmg3d , only datas on vertices are taken into account.

The keywords are the following:

SolAtVertices

SolAtEdges,

SolAtTriangles,

SolAtQuadrilaterals,

SolAtTetrahedra,

SolAtPentahedra,

SolAtHexahedra,

SolAtVertices,

SolAtVertices,

SolAtEdges,

SolAtTriangles,

SolAtQuadrilaterals,

SolAtTetrahedra,

SolAtPentahedra,

SolAtHexahedra,

Next you have to give the number of entities:

number of vertex for SolAtVertices,

number of triangles for SolAtTriangles,

number of tetrahedra SolAtTetrahedra,

etc...

Then you have to indicate the number of solution and the type of each of them:

1 for scalar solution

2 for vectorial solution

3 for tensor (only symetrical tensor are treated)

Example 1: scalar solution specifed on mesh vertices.

```
MeshVersionFormatted
1

Dimension
3

SolAtVertices
10
1 1
0.396303
0.364391
0.422417
0.456066
0.5
0.5
0.5
0.5
0.10942
0.257269

End
```

Example 2: three solutions specified on mesh tetrahedra. In this example, the first solution is a scalar one (one value per tetrahedron), the second is a vectorial one (two values) and the third is a tensorial one (three values).

```
MeshVersionFormatted
1

Dimension
2

SolAtTetrahedra
5
3 1 2 3
```

```
0.396303 0.2 0.3 0.4 0.5 0.7
0.364391 0.2 0.3 0.4 0.5 0.7
0.422417 0.2 0.3 0.4 0.5 0.7
0.456066 0.2 0.3 0.4 0.5 0.7
0.5 0.2 0.3 0.4 0.5 0.7

End
```

Example 3: vectorial solution specifed on mesh vertices.

```
MeshVersionFormatted
1

Dimension
3

SolAtVertices
5
1 2
0.396303 0.2 0.3
0.364391 0.2 0.3
0.422417 0.2 0.3
0.456066 0.2 0.3
0.5 0.2 0.3

End
```

Binary format The MESH and SOL can be written in binary. It is also based on keywords but those last are given with numbers. The correspondance table betwenn numbers and keywords are given follow:

MeshVersionFormatted	1
Dimension	3
Vertices	4
Edges	5
Triangles	6
Quadrilatérals	7
Tetrahedra	8
Pentahedra	9
Corners	13
Ridges	14
RequiredVertices	15
RequiredEdges	16
RequiredTriangles	17
RequiredQuadrilaterals	18
TangentAtEdgesVertices	19
NormalAtVertices	20
NormalAtTriangleVertices	21
NormalAtQuadrilateral	22
AngleOfCorner	23
BoundingBox	50
Tangent	59
Normal	60
TangentAtVertices	61
SolAtVertices	62

Thus the file must begins with the integer 1 which has been followed by the version number (1 for simple précision, 2 for double précision).

Next, there is the integer 3 (Dimension) then an integer corresponding to the file position at the end of the information concerning the dimension, *i.e.* after the integer indicated the dimension (so it is the number 20: an integer is stored on 4 bits and including this one, we will write five integers) then the space dimension (2 or 3).

The continuation of the file is composed by other keywords with the following convention: just after the keyword number, you have to indicate the file position at the end of the information concerning this keyword then you give this information.

Example : To write in binary the following mesh:

```
MeshVersionFormatted 2

Dimension 3

Vertices
4
0 0 0 1
1 0 1 1
0 1 1 1
1 1 1 1

Tetrahedra
1
1 2 3 4 52

End
```

you have to write in binary file the following number:

```
1 2 3 20 3 4 132 0. 0. 0. 1 1 0. 1. 1 0. 1. 1. 1 1. 1. 1. 1 8 1 2 3 4 52 54
```

Explanation :

1 : `MeshVersionFormatted`

2 : version number

3 : `Dimension`

20 : file position at the end of the information concerning the keyword `Dimension`

4 : `Vertices`

140 : file position at the end of the information concerning the keyword `Vertices`: each vertex is composed by three double precision real and an integer (reference), and a double precision real is stored on 8 bits thus each vertex information is composed by 8*3+4=28bits. There are 4 vertex so 112 bits. Then we add the integers for the keyword and the number of vertices (2*4 bits) and 20 bits for the previous fields. The result is well 140.

8 : `Tetrahedra`

168 : file position at the end of the information concerning the keyword `Tetrahedra`: each tetrahedron information is composed by 5 integers.

54 : `End`